DRAFT -- To be published in DMT'09 Proceedings
(see http://ngmdb.usgs.gov/Info/dmt/)

Improving ArcGIS workflow: Automation using Visual Basic for Applications (VBA)

By Andrew L. Wunderlich

Tectonics & Structural Geology Research Group
Department of Earth and Planetary Sciences
and Science Alliance Center of Excellence
306 Earth & Planetary Sciences Building
1412 Circle Drive
University of Tennessee
Knoxville, TN 37996-1410 USA
Telephone: (865) 974-6448
Fax: (865) 974-9326
email: gibbon@utk.edu

ABSTRACT

Many tasks associated with editing and quality checking (QCing) GIS datasets can be tedious, repetitive, and time consuming. Automation of some of these tasks, such as systematic panning around a map and semi-automated feature attribution, is very desirable. Using the Visual Basic for Applications (VBA) development environment, which is included and fully supported in ArcGIS, and the many free code resources available online, applications can be developed to automate and speed up these tasks. This paper covers: 1) some basic information about programming in ArcGIS; 2) discussion of how to plan and budget time for scripting; and 3) a list of online resources used for VBA code reference, samples, and help.

INTRODUCTION TO AUTOMATION AND SCRIPTING IN ARCGIS

The ESRI ArcGIS software package has long had the ability to be customized by its users to produce innovative solutions to all sorts of problems. Whether it is overcoming a shortfall in the functionality of the original software, automating a repetitive task, or simply preparing a template or customized layer symbology or feature representation, there is almost always a way to create a tailored solution through one of the three common ways to customize ArcGIS. From the most basic to the most advanced, they are: 1) layer definition files, styles, and map templates; 2) ModelBuilder process modeling and Python scripting; and 3) Visual Basic, C++, or Java scripting. While definition files and styles are useful for "visual automation," i.e., customization and replication of *representations* of feature symbology and labeling in map layers, it does not allow for the customization of *functions* (e.g. geoprocessing) applied to features within ArcMap. Templates go a step further and allow users to customize a set of layers that are loaded automatically each time ArcMap is started, and they also permit storage of custom toolbars and macros. The second method of customization using ModelBuilder or Python begins to expose more robust automation in ArcGIS and gives users the ability to design scripts that can be used

to automate the manipulation of features, their classes and their attributes. ModelBuilder is a component of ArcCatalog that lets the user drag-and-drop ArcToolbox functions into a window and assemble them in a flowchart-like fashion to string the output of one function to the input of the next until a final output is generated. For example, the user might need to repetitively select features from a layer based on an attribute, clip those features based on a feature in another layer, reproject those features, store them in a new feature class, and add them to the map display. Python scripting takes the power of these models a step further, allowing the user to add more advanced logic and iterative loops to process features based on more complex criteria. The third method, using Visual Basic (or VBA), C++, and Java scripting, is the most advanced and most robust form of customization. These programming languages have access to ArcObjects, which is the core of the ArcGIS application suite, and let the user create scripts that interact with the application directly as actions take place, as well as create customizations to the user interface. This paper will focus on the third segment of customization: use of VBA for creating custom tools for use in ArcMap.

ARCGIS, ARCOBJECTS, AND VBA

At the core of the ArcGIS application suite is a set of modules that provide access to all the functionality available in each component of the software package. This module set is called ArcObjects. ESRI defines ArcObjects as, "a set of platform-independent software components … that provides services to support GIS applications on the desktop" (ESRI Developer Network, 2009a). These software modules were developed within a framework that allows them to be used across platforms (Windows, Linux, etc.), across applications (ArcMap, ArcCatalog, etc.), and have their properties accessed by various programming languages. For more information about ArcObjects and ArcGIS software architecture, see the ArcGIS Resource Center website (http://resources.esri.com/arcgisdesktop/).

While there are many advantages to having such an open, modular architecture, one of particular interest is the ability to utilize these powerful modules at several levels of programming expertise. Since many GIS users have scientific backgrounds and are not native users, it can be a struggle to get over the steep learning curve of the GIS application itself, let alone getting into programming and application customization. But ArcGIS has an integrated programming environment in which the full power of ArcObjects is available: Visual Basic for Applications (VBA). As ESRI states: "It [VBA] provides a programming environment, the Visual Basic Editor (VBE), which lets you write a Visual Basic (VB) macro, then debug and test it right away in ArcMap or ArcCatalog. A macro can integrate some or all of Visual Basic's functionality, such as using message boxes for input, with the extensive object library that ArcMap and ArcCatalog expose" (ESRI Developer Network, 2009b).

Visual Basic is probably one of the easiest programming languages to learn, due to its relatively intuitive syntax and lack of visual clutter in the code. The VB editor (VBE) has many resources built in to help a novice user to understand the code, including auto-complete when typing code for function properties (to avoid accessing inappropriate properties), as well as an object browser that explains object relationships and gives definitions of object properties. VBA can access global variables within the document, application, and operating system so the user can create hooks into the application that trigger actions based on what is taking place within the user environment. Users can create custom user interface forms visually using many of the same

components available in larger, more robust programming environments such as VB .NET and Visual C++. Also, close relationships of VBA with VB 6 and VB .NET allow relatively easy migration of VBA code to VB Dynamic Link Library projects in those environments. The Visual Basic Editor interface and run-time debugging tools quickly and easily test and debug macros inside ArcMap and ArcCatalog without having to worry about compiling and testing code in another environment. Finally, the single greatest advantage to VBA is the availability of thousands of code samples in the help system and on the web that are ready to be cut, pasted, and run within the VBE environment without extensive programming experience on the part of the user. For more information about VBA and its functionality, see "Getting started with VBA" in the ArcGIS Desktop Help (http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Getting_started_with_VBA).

SCRIPT DESIGN: GOALS AND BUDGETING TIME

A common misconception about programming is that it requires a computer science degree, and many years of training, for successful use. Another misconception is that it takes a lot of time to create a customization. The truth is that with the availability of so many examples and "canned" code on the web, and in the help documentation of VBA, anyone can quickly create simple scripts to help automate a process in ArcGIS. The key things to remember are to *keep scripts simple* and that automations *save time in the long run*. Scripts do not need to be complex. They do not need to be polished bits of code with all the fail-safes that professionally-coded applications have. They only need to be as functional as you need them to be, and once you get them working, you can continue to improve them and add more functionality or more robust error handling for use by a wider audience.

When tackling a scripting project, the first step is to clearly identify the goals of the automation. Again, this is a process of identifying the bare-bones functionality, then deciding what (if any) additional fail-safes or bonus features should be added if time allows. In most cases, the steps in the automation are exactly the same as if you were to do the process manually. Consider the following scenario from a script I designed to help with the attribution of features: structural symbols were digitized from a georeferenced scan of a paper map and needed to have the dip value and symbol type attributes added to each feature. If this were to be done manually, the process would go something like this, assuming the features had already been created: (1) start an edit session; (2) select the features that need attribution (select all or a subset); (3) open the attribute table or the Editor Attributes window and pan/zoom to the first feature; (4) type the value for the dip or the symbol type code into the appropriate field and press <Enter>; (5) pan/zoom to the next feature; and (6) repeat steps 4-5 until all features have been attributed (Figure 1). This list of actions becomes the basis for the necessary functions of the script and helps define the keywords for searching for the code snippets that will be needed to create the script, which will be discussed in the next section.

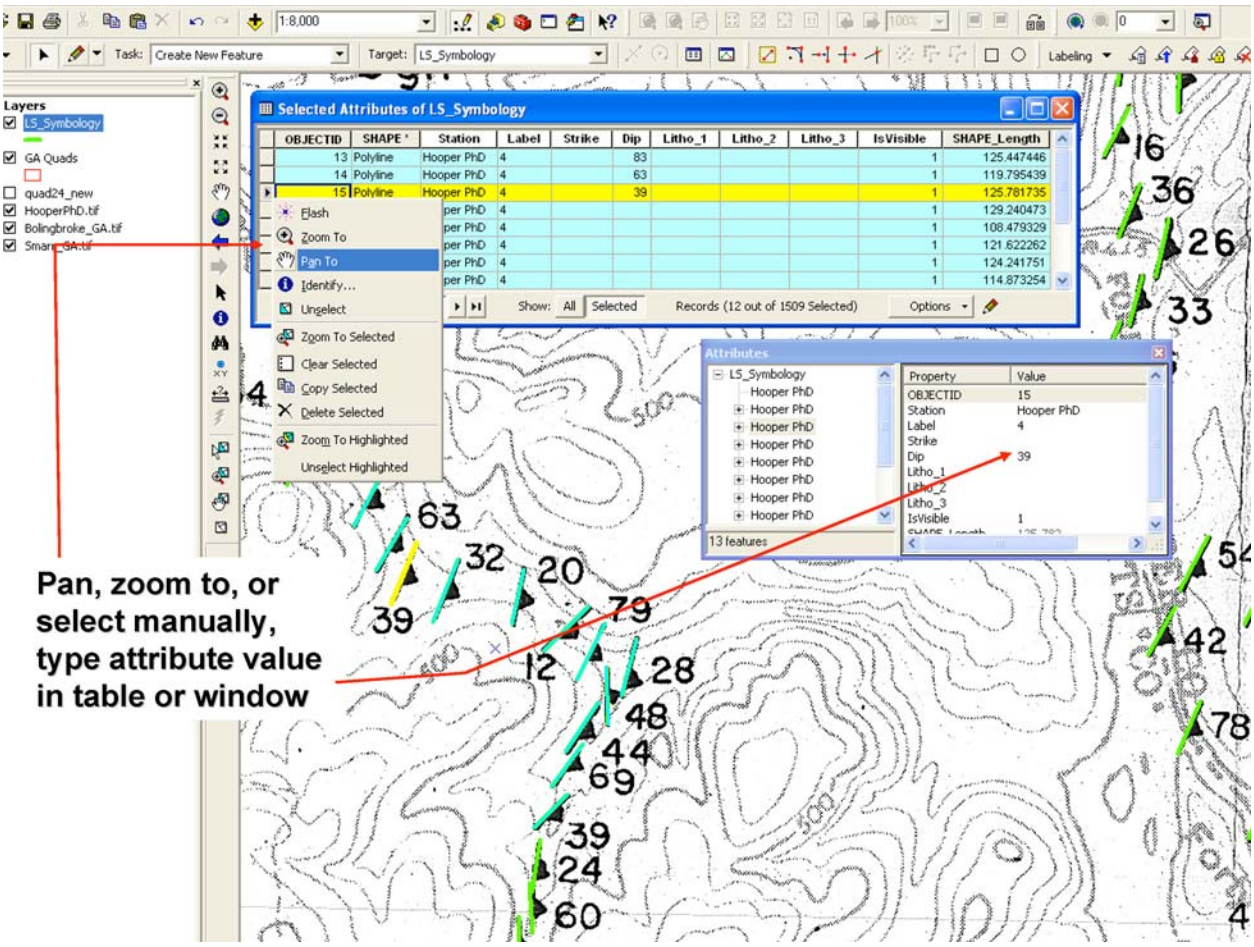

Figure 1. Screenshot of ArcMap displaying an excerpt of a geologic map scanned from a Mylar sheet. The layer attribute table as well as the Editor feature attribute window are open. Selected features can have attribute values added manually through either interface. The "Attribute Features" script seeks to automate this laborious data entry process.

There will undoubtedly be a need to justify the time spent on creating a customization. To do so, create a simple matrix similar to that in Table 1 as an aid in presenting your case to supervisors or management. Taking the example scenario outlined in the previous paragraph, I attributed a few dozen features using the "by hand" method in order to get a sense of how much time it would take to perform the attribution manually (row 1). In my experience with automating processes, the time savings can be from 50-90% depending on the function. A safe figure to use when estimating the time savings is probably 50% (row 2). This should provide a reasonable estimate of the time savings over many iterations of a common task that you wish to automate. I also calculated the actual savings after the script was completed and used to attribute all the features on a map (row 3). In this particular scenario, the actual time savings per feature were quite substantial, around 80% compared to the manual method, not counting the time taken to put the code together. Considering the time taken to create the basic version of the script (about one working day), the savings from converting only one map of this type does not appear significant since the time would be almost equal. However, utilizing this script on just two maps would probably justify the time spent on the automation, and the time saved on ten maps would

be even more significant. Also, it is important to remember that, if you are a newcomer to programming of any kind, taking a couple of hours every other day over the course of a few weeks to familiarize yourself with some basics of VBA will really help to speed script creation.

	Time per feature (sec)	Features to process on one map (avg)		Time to process 1 map (hrs)	Total Time (inc. 8 hrs dev.)	Total Time for 10 maps (hrs)
By hand (no auto)	25	1500	25s x 1500 = 37500s / 3600s/hr =	10.42	n/a	104.17
Est. w/ auto (50% of "by hand")	12.5	1500	12.5s x 1500 = 18750s / 3600s/hr =	5.21	13.21	60.08
Actual (timed w/ auto)	4.5	1500	4.5s x 1500 = 6750s / 3600s/hr =	1.88	9.88	26.75
			Time needed to develop basic automation =		8	hrs

Table 1. Matrix for comparing time estimates for a task in ArcMap to help determine the benefit from development of an automation.

SCRIPT SETUP, SEARCHING AND ASSEMBLING CODE

The best resource for learning to code is the ArcGIS Desktop Help (http://webhelp.esri.com). Under the topic "Writing macros using VBA", the section entitled "Sample VBA code" has ten scripting samples that are designed to help the novice user understand some of the important scripting tasks, including adding layers to a map and calculating values for a field in an attribute table. These samples have detailed instructions on how and where to put the code in the VBE and how to call the script in your ArcMap session. Once you have familiarized yourself with the VBE and code samples, and how they work, you can begin to search for the code samples that you need to complete your automation goal.

After identifying the basic requirements for your script (as outlined in the previous section), begin your coding by searching on the ESRI Developer Network website (http://edn.esri.com) or the ESRI Support website (http://support.esri.com) using keywords for the functions you are trying to automate. Code samples that can accomplish these tasks can be found using the example of the semi-automatic attribution of features, searching on keywords such as "select features programmatically," "store attribute," "zoom to feature," and "loop through selected features." It is often the case that the code samples returned for such searches are much more complex than necessary. Sometimes the code will have to be read through and only the parts that pertain to the task at hand need to be extracted. This process can be time consuming, but as you become more familiar with the syntax and form of the code, looking for the sections you need becomes much easier. Also, an additional advantage to many of the samples available from ESRI Developer Network in particular, is that the code is well described with internal comments that explain what the code does, section by section. Remember, you don't have to completely understand why a piece of code works as long as you can organize the code sections logically to execute and obtain a suitable outcome. The key is to use existing scripts and modify them. This is an excellent way to learn VBA (or any programming language): take code that is already written and play with it, modify it, and learn how it works through trial-and-error testing.

TESTING SCRIPTS: TIPS AND TRICKS

Once pieces of code have been found, assemble them in the VBE and begin the process of testing and debugging the code. It is important to note that testing should *never* be done on *anything* but sample datasets or copies of datasets. Until you are confident that your code is stable and will work properly, anything could happen, including lock-ups, crashes, or corruption to datasets, so save your work often. Also, never run multiple sessions of ArcMap when scripting with VBA. This will prevent problems with the Normal.mxt (the default template initialized each time ArcMap starts), which stores code and user form modules.

In most cases, running code that contains errors (which is inevitable) will cause the VBE to stop execution of the code and present the user with an error message. The error message offers the option to view the offending line and help to fix the problem. There may be cases where the error is not a syntax issue *per se*, but one of unexpected behavior or poor performance, as with a non-exiting loop or use of an improper cursor. In a case like this, the user can stop code execution at any time by pressing <Ctrl> + <Pause/Break>. Sometimes, errors are not identified directly by the line on which they occur. The best way to trap errors of an ambiguous nature may be to execute the code line by line or to establish breakpoints in the code to step through section by section. The Immediate Window in the VBE and/or pop-up message boxes are another way to display variables being used as the code executes, or to signal the completion of a section of code:

```
' Give a value to an object:
Dim sValue As String
sValue = "Hello world!"
' Display the value to the user using message box:
MsgBox "The value of sValue is: " & sValue
' Display the value to the user using the Immediate Window:
Debug.Print "The value of sValue is: " & sValue
```

This method can help identify problems by showing when a variable that is needed returns a null value (“”) or the word “Nothing”.

Once code has been tested and seems to work well within your specifications, save the working code to a text file or export the code module from the VBE to a file. As code becomes more complex, errors can be introduced by the addition of, or changes to, code that cannot be undone, and reversion may be necessary. *Never* delete lines of code unless you are sure they are not necessary. Instead, use comments to “hide” code from being executed. Comments are simply a single quote character “'” that precedes any part of a line of code and can be used to make the VBE ignore a line:

```
' Execute this line:
Set pDataStats = New esriGeoDatabase.DataStatistics 'after comment
' Not this one:
'Set pDataStats = esriGeoDatabase.DataStatistics
```

This can be very helpful in testing different syntax without deleting anything in case a mistake is made. Only at the end of a project do I remove lines as part of the project cleanup and preparation to post the code for public dissemination. Comments can also be used to describe a

function or explain the use of a certain object or variable, making code easier to understand and be used by others (Figure 2).

Notice that the code in the VBE window shown in Figure 2 is colorized to improve the readability. By default, comments will be green, keywords will be blue, errors will be red, and normal and parameter text will be black. These colors can be changed by the user, but I do not recommended it since these are standard, and many of the code samples from the web and ESRI Developer Network are colored in this way.

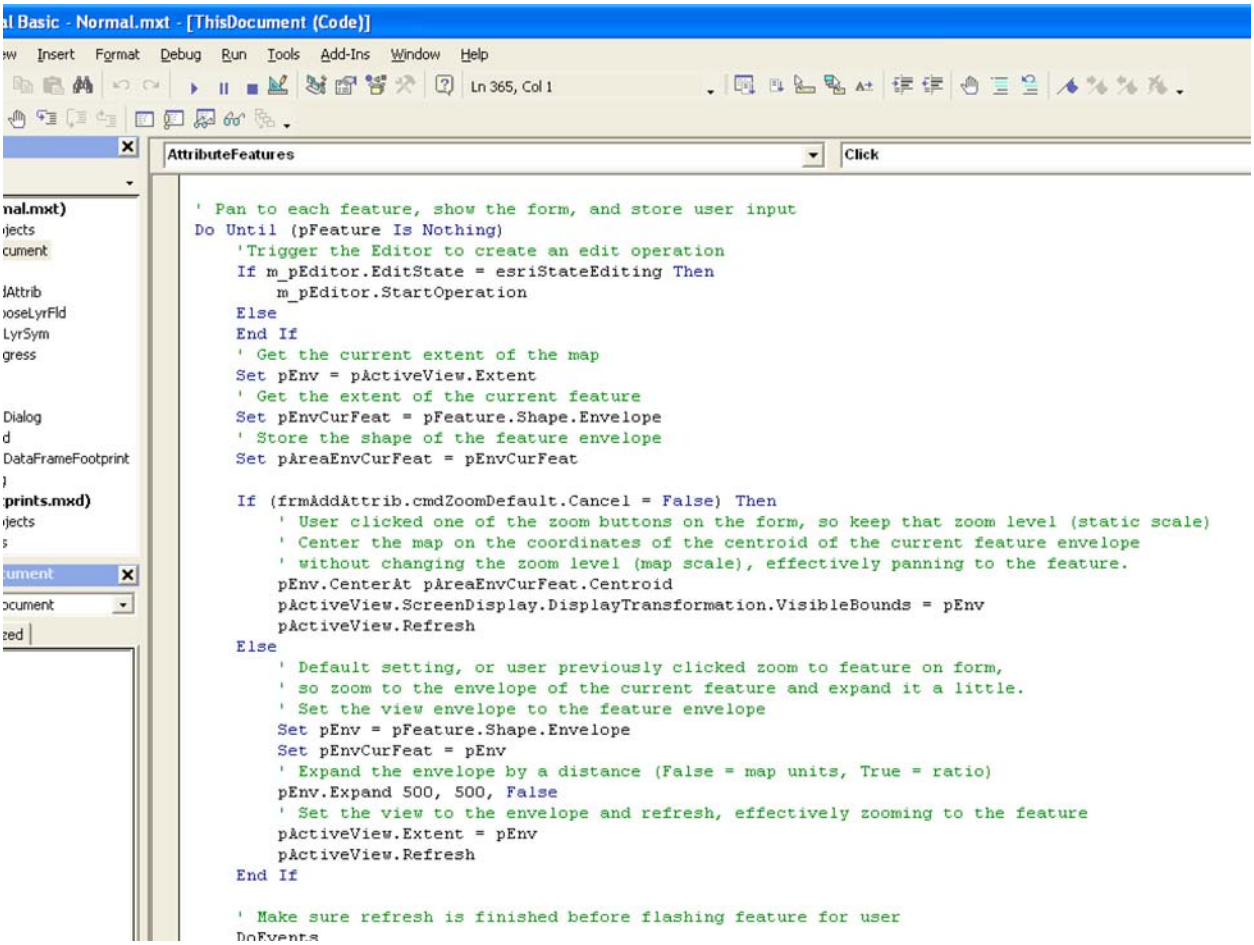

Figure 2. Screenshot of the Visual Basic Editor code window. Notice comments in the code (in green), marked by "'", improve the readability of the code and make it easier to understand the functions.

FINISHING CODE: FINAL CONSIDERATIONS

When your script is complete there are a few things that you can consider adding to make your code more robust and more polished, especially if you are considering posting your code samples to the ESRI Resource Center Code Galleries or ArcScripts. One important feature for all code is the ability to gracefully quit and notify the user when an error occurs. Even if you test your code a thousand times with no errors, distribute the code and someone will surely find a

way to make it crash. A simple error handler is not hard to add and makes your code able to deal with errors and exceptions in a more professional way:

```
Public Sub StartGeoMapEditSession_Click()
' Put this line of code after the first line of the Sub or Function:
On Error GoTo ErrHandler

' MAIN BODY OF CODE GOES HERE

' Just before End Sub or End Function:
Exit Sub  ' or "Exit Function"

ErrHandler:
MsgBox "Error Number: " & Err.Number & vbCr & "Error Source: " & Err.Source &
vbCr & "Error Description: " & Err.Description & "." & vbCr & vbCr &
"Operation canceled!!", vbCritical, "ArcMap"

'Now comes End Sub or End Function
End Sub  ' or "End Function"
```

With this code in your script, if there is an error while running the Sub or Function, the operation will be canceled and a message box that identifies the error number—the error source module—and the error description (if applicable) will be displayed.

Another nice feature to add to your code is a comment section at the beginning of the sub or function that identifies who created it, describes what the function does, when/what updates have been made, etc.:

```
Private Sub CreateDataFrameFootprint_Click()

' Created by Andrew L. Wunderlich
' September 10, 2007
' Updated:
' December, 2007 - added data frame rotation
' June, 2008 - fixed scale number problem
' May, 2009 - added densification routine
' ================================
' This tool converts the data frame
' envelope to a polygon and exports
' it to a shapefile with user-
' specified name and location and
' adds it to the map document.
' The tool also takes into account
' Data Frame rotation, if any, and
' rotates the polygon to match.
' The output shapefile contains three
' fields: the map document name,
' the map scale, and the rotation
' in degrees (counterclockwise).
```

As you work on your own projects and search and retrieve code samples from the Web, you will certainly find other things that you will want to add to your code to make it more robust, user friendly, and functional. Try different things and be creative: a solution to almost any automation goal can be achieved by searching the resources and using the methods outlined here.

CODING RESOURCES

These resources help you create custom code:

Getting started with VBA:

- "Getting started with VBA" in the ArcGIS Desktop Help
http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Getting_started_with_VBA

- "Sample VBA Code" in the ArcGIS Desktop Help
http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Sample_VBA_code

- "Customizing ArcGIS desktop with VBA"
http://resources.esri.com/help/9.3/arcgisdesktop/com/vba_start.htm

- ArcGIS Web Help – general topics
http://webhelp.esri.com

ESRI Developer Network:

- Homepage for scripting with ArcGIS 9.2 (and earlier)
http://edn.esri.com

- Code Exchange – find code samples and documentation
http://edn.esri.com/index.cfm?fa=codeExch.gateway

ESRI Resource Center:

- ArcGIS 9.3 (and later) resources for developers
http://resources.esri.com/arcgisdesktop/index.cfm?fa=forDevelopers

ESRI Support Center and ArcScripts:

- ESRI Support Center – search for help with solutions to automation problems. User can create an ESRI Global Account to post questions, watch threads, and post solutions to others' problems. Highly recommended!
http://support.esri.com

- ArcScripts – homepage for user community script posting and exchange. Code for scripts I presented in my PowerPoint at the DMT '09 meeting, including the Attribute Features script, can be downloaded. Search with keyword "Wunderlich."
http://arcscripts.esri.com

ACKNOWLEDGEMENTS

Support by Dr. Robert D. Hatcher, Jr., the University of Tennessee Science Alliance Centers of Excellence, and the USGS NCGMP through Grant Number 08HQGR0098 is greatly appreciated.

REFERENCES

ESRI Developer Network, 2009a, What is ArcObjects?: ArcObjects: Foundation of ArcGIS, accessed at http://edndoc.esri.com/arcobjects/9.2/CPP_VB6_VBA_VCPP_Doc/shared/ao_foundation/what_is_ao.htm

ESRI Developer Network, 2009b, How can you customize ArcGIS Desktop?: Getting started developing for ArcGIS Desktop, accessed at http://edndoc.esri.com/arcobjects/9.2/CPP_VB6_VBA_VCPP_Doc/COM/VBA/what_develop_dtop.htm